CMSC/Math 456: Cryptography (Fall 2023)

Lecture 3

Daniel Gottesman

Administrative

First problem set is out. Due on Thursday, Sep. 7, noon (i.e., before the start of class) on Gradescope.

Reminder: Extensions need prior approval and valid reason.

This one should not be too challenging, but the problem sets will get harder.

The textbook is on reserve with the library and can be checked out for 4 hours at a time.

If you are reading the slides before we get to a point in the lecture and see a Vote, stop and think about your answer before reading further.

This class is being recorded

Definition of Encryption

Definition: A private-key encryption protocol is a set of three probabilistic algorithms (Gen, Enc, Dec).

Gen is the key generation algorithm. It takes as input s, the security parameter, and outputs a key $k \in \{0,1\}^*$.

Enc is the encryption algorithm. It takes as input k and a plaintext or message $m \in \{0,1\}^*$ and outputs a ciphertext $c \in \{0,1\}^*$.

Dec is the decryption algorithm. It takes as input k and c and outputs some $m' \in \{0,1\}^*$.

An encryption protocol is correct if

$$Dec(k, Enc(k, m)) = m$$

Unless otherwise stated, assume that Gen(n) chooses a random bit string of length s. Note that there may be some restrictions on the allowed space of messages (e.g., length).

One-Time Pad

The one-time pad is defined as follows:

The security parameter s should be chosen to be equal to the message length to be used.

Gen: Choose uniformly random bit string k of length s.

Enc: Acts on message m of length s as $Enc(k, m) = m \oplus k$.

Dec: Acts on ciphertext c of length s as $Dec(k, c) = c \oplus k$.

Definition of Perfect Secrecy

Definition A: An encryption protocol (Enc, Dec) provides perfect secrecy if for any distribution M of valid messages and any $m \in M$, ciphertext c such that $\Pr(Enc(k,m) = c) \neq 0$,

$$\Pr(M = m \mid C = c) = \Pr(M = m)$$

averaged over keys k and randomness in Enc and Dec.

Alternatively,

Definition B: An encryption protocol (Enc, Dec) provides perfect secrecy if for any pair of valid messages m_1, m_2, and any ciphertext c,

$$\Pr(Enc(k, m_1) = c) = \Pr(Enc(k, m_2) = c)$$

with probability averaged over keys k and randomness in Enc and Dec.

These are known as soundness conditions.

Meaning of Definitions

Definition A, $\Pr(M = m \mid C = c) = \Pr(M = m)$, says that Eve's best guess about the message after seeing the ciphertext is the same as her best guess before: She hasn't learned any information.

Definition B, $\Pr(Enc(k, m_1) = c) = \Pr(Enc(k, m_2) = c)$, says that any given ciphertext is equally likely to result whether the message is m_1 or m_2. This definition is often easier to work with, but it might be less obvious what it has to do with secrecy.

> Definition B basically says that if Eve is trying to choose between the two messages, seeing the ciphertext doesn't help her.

Should we choose definition A or definition B? Vote.

Equivalence of Definitions

Definition A ⟺ Definition B

Proof: Relies on Bayes' theorem

$$\Pr(M = m \mid C = c) = \frac{\Pr(C = c \mid M = m)\Pr(M = m)}{\Pr(C = c)}$$

Equivalence of Definitions

Definition A ⟺ Definition B

Proof: Relies on Bayes' theorem

$$\Pr(M = m \mid C = c) = \frac{\Pr(C = c \mid M = m)\Pr(M = m)}{\Pr(C = c)}$$

A ⇒ B: Consider a distribution of messages that contains both m_1 and m_2 as possibilities. Then definition A says that

$$\Pr(M = m_i \mid C = c) = \Pr(M = m_i)$$

(unless $\Pr(C = c) = 0$). By Bayes' theorem, this implies that

$$\Pr(C = c \mid M = m_1) = \Pr(C = c) = \Pr(C = c \mid M = m_2)$$

Equivalence of Definitions

Definition A ⟺ Definition B

Proof: Relies on Bayes' theorem

$$\Pr(M = m \mid C = c) = \frac{\Pr(C = c \mid M = m)\Pr(M = m)}{\Pr(C = c)}$$

A ⟹ B: Consider a distribution of messages that contains both m_1 and m_2 as possibilities. Then definition A says that

$$\Pr(M = m_i \mid C = c) = \Pr(M = m_i)$$

(unless $\Pr(C = c) = 0$). By Bayes' theorem, this implies that

$$\Pr(C = c \mid M = m_1) = \Pr(C = c) = \Pr(C = c \mid M = m_2)$$

But $\Pr(C = c \mid M = m_i) = \Pr(Enc(k, m_i) = c)$ by the definition of Enc, so we get definition B.

Equivalence of Definitions

Definition A $\Longleftrightarrow$ Definition B

Proof: Relies on Bayes' theorem

$$\Pr(M = m \mid C = c) = \frac{\Pr(C = c \mid M = m)\Pr(M = m)}{\Pr(C = c)}$$

A $\Rightarrow$ B: Consider a distribution of messages that contains both m_1 and m_2 as possibilities. Then definition A says that

$$\Pr(M = m_i \mid C = c) = \Pr(M = m_i)$$

(unless $\Pr(C = c) = 0$). By Bayes' theorem, this implies that

$$\Pr(C = c \mid M = m_1) = \Pr(C = c) = \Pr(C = c \mid M = m_2)$$

But $\Pr(C = c \mid M = m_i) = \Pr(Enc(k, m_i) = c)$ by the definition of Enc, so we get definition B.

If $\Pr(C = c) = 0$, then $\Pr(Enc(k, m_i) = c) = 0$ for both i.

Equivalence of Definitions, cont.

B ⇒ A: For any two messages m_1 and m_2, we have

$$\Pr(C = c \mid M = m_1) = \Pr(Enc(k, m_1) = c)$$
$$= \Pr(Enc(k, m_2) = c) = \Pr(C = c \mid M = m_2)$$

This class is being recorded

Equivalence of Definitions, cont.

B ⇒ A: For any two messages m_1 and m_2, we have

$$\Pr(C = c \mid M = m_1) = \Pr(Enc(k, m_1) = c)$$
$$= \Pr(Enc(k, m_2) = c) = \Pr(C = c \mid M = m_2)$$

This implies that $\Pr(C = c \mid M = m) = P$, with P a constant independent of m. Then

$$\Pr(C = c) = \sum_{m \in M} \Pr(C = c \mid M = m)\Pr(M = m) = P \sum_{m \in M} \Pr(M = m) = P$$

using the fact that probabilities sum to 1.

Equivalence of Definitions, cont.

B ⇨ A: For any two messages m_1 and m_2, we have

$$\Pr(C = c \mid M = m_1) = \Pr(Enc(k, m_1) = c)$$
$$= \Pr(Enc(k, m_2) = c) = \Pr(C = c \mid M = m_2)$$

This implies that $\Pr(C = c \mid M = m) = P$, with P a constant independent of m. Then

$$\Pr(C = c) = \sum_{m \in M} \Pr(C = c \mid M = m)\Pr(M = m) = P \sum_{m \in M} \Pr(M = m) = P$$

using the fact that probabilities sum to 1.

We get $\Pr(C = c) = \Pr(C = c \mid M = m)$, and if we plug this into Bayes' theorem, we get definition A:

$$\Pr(M = m \mid C = c) = \frac{\Pr(C = c \mid M = m)\Pr(M = m)}{\Pr(C = c)}$$

Perfect Secrecy of One-Time Pad

Theorem: The one-time pad has perfect secrecy.

Proof: We will use definition B.

Let us calculate $\Pr(Enc(k,m) = c)$ for some m and c, averaged over keys k.

Perfect Secrecy of One-Time Pad

Theorem: The one-time pad has perfect secrecy.

Proof: We will use definition B.

Let us calculate $\Pr(Enc(k,m) = c)$ for some m and c, averaged over keys k.

Given *any* m and c, then $Enc(k,m) = c$ iff $k = m \oplus c$. Therefore, there is exactly one value of k for which $Enc(k,m) = c$. Thus,

$$\Pr(Enc(k,m) = c) = \frac{1}{2^s}$$

Perfect Secrecy of One-Time Pad

Theorem: The one-time pad has perfect secrecy.

Proof: We will use definition B.

Let us calculate $\Pr(Enc(k, m) = c)$ for some m and c, averaged over keys k.

Given *any* m and c, then $Enc(k, m) = c$ iff $k = m \oplus c$. Therefore, there is exactly one value of k for which $Enc(k, m) = c$. Thus,

$$\Pr(Enc(k, m) = c) = \frac{1}{2^s}$$

This is true regardless of m, so in particular, for any m_1, m_2,

$$\Pr(Enc(k, m_1) = c) = \Pr(Enc(k, m_2) = c) = \frac{1}{2^s}$$

What Does Perfect Security Imply?

Now we have proven the one-time pad is perfectly secure, so we are done with encryption.

What Does Perfect Security Imply?

Now we have proven the one-time pad is perfectly secure, so we are done with encryption.

Well, maybe not quite …

The one-time pad has a key as long as the message. That can be inconvenient. And why is it called the one-time pad?

What if we use the same key twice for different messages?

What Does Perfect Security Imply?

Now we have proven the one-time pad is perfectly secure, so we are done with encryption.

Well, maybe not quite …

The one-time pad has a key as long as the message. That can be inconvenient. And why is it called the one-time pad?

What if we use the same key twice for different messages?

$$c_1 = m_1 \oplus k$$
$$c_2 = m_2 \oplus k$$

Add these:

$$c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2$$

This gives us a new ciphertext $c_1 \oplus c_2$ which is effectively one message encrypted using the other. This is like using a book with the Vigenère cipher: It is vulnerable to frequency analysis.

Limits of Security Definitions

The definition of perfect secrecy assumes only one message. If you are going beyond it, e.g. by sending two messages, then it does not provide any security guarantee.

This matters: For instance, during the Cold War, Soviet spies in the U.S. were using the one-time pad to communicate, but they had some duplicated pages in their codebooks, causing them to use some keys twice. The U.S. was able to break some of their messages (the Venona project), helping to uncover many spies.

But security definitions and proofs help to delineate exactly what a cryptographic protocol can do and what it can't do.

The need to replace the key after each message is extremely inconvenient and limits the one-time pad to the applications needing highest security.

Other Limits of the One-Time Pad

Perhaps we should think carefully about the one-time pad to see if there are any other hidden limitations.

Other Limits of the One-Time Pad

Perhaps we should think carefully about the one-time pad to see if there are any other hidden limitations.

Perfect secrecy says Eve cannot gain any information about the distribution of messages of the same length. It says nothing about messages of different lengths.

Sometimes this doesn't matter. Sometimes it does. (Can pad messages to make them the same length.)

Other Limits of the One-Time Pad

Perhaps we should think carefully about the one-time pad to see if there are any other hidden limitations.

Perfect secrecy says Eve cannot gain any information about the distribution of messages of the same length. It says nothing about messages of different lengths.

Sometimes this doesn't matter. Sometimes it does. (Can pad messages to make them the same length.)

If the key happens to be k = 000…0, the messages are being sent unencrypted! Maybe we should exclude this key from Gen?

Vote: If we do, is the result stronger/weaker/the same?

Other Limits of the One-Time Pad

Perhaps we should think carefully about the one-time pad to see if there are any other hidden limitations.

Perfect secrecy says Eve cannot gain any information about the distribution of messages of the same length. It says nothing about messages of different lengths.

Sometimes this doesn't matter. Sometimes it does. (Can pad messages to make them the same length.)

If the key happens to be k = 000…0, the messages are being sent unencrypted! Maybe we should exclude this key from Gen?

Vote: If we do, is the result stronger/weaker/the same?

Answer: Weaker! Why? The scheme is no longer perfectly secret: If Eve sees ciphertext c, she now knows $m \neq c$, which she might not have known before.

Shorter Key Lengths?

We would like to make the one-time pad more convenient to use. One way to do this would be to find an encryption protocol with perfect secrecy and a shorter key.

Theorem: Any protocol with perfect secrecy must have as many possible keys as possible messages.

Shorter Key Lengths?

We would like to make the one-time pad more convenient to use. One way to do this would be to find an encryption protocol with perfect secrecy and a shorter key.

Theorem: Any protocol with perfect secrecy must have as many possible keys as possible messages.

Proof: Consider a fixed ciphertext c. For all m_1 and m_2,

$$\Pr(Enc(k, m_1) = c) = \Pr(Enc(k, m_2) = c)$$

Shorter Key Lengths?

We would like to make the one-time pad more convenient to use. One way to do this would be to find an encryption protocol with perfect secrecy and a shorter key.

Theorem: Any protocol with perfect secrecy must have as many possible keys as possible messages.

Proof: Consider a fixed ciphertext c. For all m_1 and m_2,

$$\Pr(Enc(k, m_1) = c) = \Pr(Enc(k, m_2) = c)$$

In particular, there must be some key k_1 and choice of randomness such that $Enc(k_1, m_1) = c$ and some key k_2 and randomness such that $Enc(k_2, m_2) = c$. By correctness,

$$Dec(k_i, c) = m_i$$

This class is being recorded

Shorter Key Lengths?

We would like to make the one-time pad more convenient to use. One way to do this would be to find an encryption protocol with perfect secrecy and a shorter key.

Theorem: Any protocol with perfect secrecy must have as many possible keys as possible messages.

Proof: Consider a fixed ciphertext c. For all m_1 and m_2,

$$\Pr(Enc(k, m_1) = c) = \Pr(Enc(k, m_2) = c)$$

In particular, there must be some key k_1 and choice of randomness such that $Enc(k_1, m_1) = c$ and some key k_2 and randomness such that $Enc(k_2, m_2) = c$. By correctness,

$$Dec(k_i, c) = m_i$$

which implies that $k_1 \neq k_2$. But this is true for all pairs m_1 and m_2, so there must be at least one key for each message.

Beyond Perfect Secrecy

To do better, we must give up some of the conditions of perfect secrecy. We don't need that Eve's gain of information is 0, provided it is very small. Unfortunately, by itself, this is not enough for much improvement.

> One idea is to work on the key. In the Vigenère cipher with a book as a key, Alice and Bob's short key gave them a recipe for how to generate a much longer sequence of shifts for the encryption algorithm by looking at the book. Unfortunately, that approach was still vulnerable to frequency analysis.

But what if we used some other source of bits that *looks* more random than a book even though it is not?

> What counts as looking random, and how can we be sure Eve doesn't have a way of analyzing our source of bits that reveals patterns that can be used to decrypt?

Threat Model

We need to precisely define Eve's capabilities and what she is trying to. We shouldn't assume how she will attack our protocol, since she might think of a different approach.

- Be precise: Define Eve's capabilities precisely, so that we can evaluate if our assumption about them is satisfied or not.
- Be conservative: It is safer to assume Eve has more power than she actually has rather than to assume she has less power than she actually has.
- Make (only) necessary assumptions: There may be some assumptions we need to make in order to have any security at all. Minimize these, but since they are critical, try to make sure they are true.

This analysis will give us the threat model.

Eve's Capabilities

Eve has the following capabilities and limits:

- Eve cannot access Alice's or Bob's computer. (Necessary; otherwise the encryption won't help.)
- Eve does not know the key.
- Eve does know the protocol being used. (Kerckhoff's Pr.)
- Eve has large but limited computational power.
- Eve can read the ciphertext of the transmitted message.
  - ✦ Or the ciphertext of multiple messages.
  - ✦ Or knows both the ciphertext and plaintext of multiple messages before this one.
  - ✦ Or chooses the plaintext of multiple messages and sees the resulting ciphertexts. (Chosen plaintext attack)
  - ✦ Or …

Eve's Capabilities

Eve has the following capabilities and limits:

- Eve cannot access Alice's or Bob's computer. (Necessary; otherwise the encryption won't help.)
- Eve does not know the key.
- Eve does know the protocol being used. (Kerckhoff's Pr.)
- Eve has large but limited computational power.
- Eve can read the ciphertext of the transmitted message.
  - ✦ Or the ciphertext of multiple messages.
  - ✦ Or knows both the ciphertext and plaintext of multiple messages before this one.
  - ✦ Or chooses the plaintext of multiple messages and sees the resulting ciphertexts. (Chosen plaintext attack)
  - ✦ Or …

What Is "Limited" Computation?

We need a precise definition of what it means for Eve to have limited computational power.

What Is "Limited" Computation?

We need a precise definition of what it means for Eve to have limited computational power.

- Limit Eve's computational time, e.g.: at most 10^6 years.

  This is what we are most interested in in practice; but it is very vulnerable to improvements in computer technology.

What Is "Limited" Computation?

We need a precise definition of what it means for Eve to have limited computational power.

- Limit Eve's computational time, e.g.: at most 10^6 years.

  This is what we are most interested in in practice; but it is very vulnerable to improvements in computer technology.

- Limit Eve's computational steps.

  This is more robust against improvements in computers, but different architectures can still require a different number of steps; and what if we increase the security parameter?

What Is "Limited" Computation?

We need a precise definition of what it means for Eve to have limited computational power.

- Limit Eve's computational time, e.g.: at most 10^6 years.

  This is what we are most interested in in practice; but it is very vulnerable to improvements in computer technology.

- Limit Eve's computational steps.

  This is more robust against improvements in computers, but different architectures can still require a different number of steps; and what if we increase the security parameter?

- Limit the rate of asymptotic scaling of Eve's number of computational steps as we increase the security parameter.

Big-O Notation

We quantify asymptotic number of steps by big-O notation, e.g. $O(s^2)$, $O(2^s)$.

Definition: We say a function $f(s) = O(g(s))$ if there exist constants C, s_0 such that $f(s) \leq Cg(s)$ for all $s > s_0$.

Big-O Notation

We quantify asymptotic number of steps by big-O notation, e.g. $O(s^2), O(2^s)$.

> Definition: We say a function $f(s) = O(g(s))$ if there exist constants C, s_0 such that $f(s) \le Cg(s)$ for all $s > s_0$.

Why? This expresses the leading-order behavior up to a constant factor. For large **s**, the leading order behavior always dominates: $s^5 + 100s^3 = O(s^5)$. And $s^5 > 100s^3$ whenever $s > 10$. The rate of growth of larger functions rapidly overcomes even large constants.

Big-O Notation

We quantify asymptotic number of steps by big-O notation, e.g. $O(s^2)$, $O(2^s)$.

> Definition: We say a function $f(s) = O(g(s))$ if there exist constants C, s_0 such that $f(s) \leq Cg(s)$ for all $s > s_0$.

Why? This expresses the leading-order behavior up to a constant factor. For large s, the leading order behavior always dominates: $s^5 + 100s^3 = O(s^5)$. And $s^5 > 100s^3$ whenever $s > 10$. The rate of growth of larger functions rapidly overcomes even large constants.

In particular, for large enough s, 2^s is always bigger than As^c for any constants A and c. For instance,

$$A = 1000, c = 10: 2^s > 1000s^{10} \text{ when } s \geq 72.$$

Exponential wins out over polynomial even for modest size s.

Polynomial Time

In complexity theory, we like to categorize functions as polynomial time vs. exponential time.

- Different models of computation can give different numbers of computational steps to compute the same function, but they only seem to differ by polynomial factors (excepting quantum computation).
- The set of polynomial functions is closed under most operations: addition, multiplication, composition. This means that the property of being ``polynomial'' is robust under a variety of different transformations. For instance, a polynomial number of calls to a subroutine that itself takes polynomial time is also polynomial.

We say that a computation that can be completed in polynomial time as a function of the size of the input is efficient. In this course, I will generally allow probabilistic algorithms.

Negligible Probabilities

We also need to consider how much information we will allow Eve to learn. Usually this manifests as a probability, for instance as an increase in the probability that Eve can correctly guess the message.

> If we choose polynomial time as the measure of efficiency and assume everything that Eve does should be in polynomial time, then what probability ϵ should we accept for Eve to do something she is not supposed to?

If Eve achieves her goal with probability ϵ in one attempt and she can try f(s) times, then we need that $f(s)\epsilon$ is small. Therefore, we call a function g(s) negligible if

$$f(s)g(s) < 1$$

for *all* polynomials f(s) (and $s > s_0$ for some constant s_0).

Example: $2^{-\sqrt{s}}$ and $s^{-\log s}$ are both negligible functions.

Drawbacks of Big-O

This approach to characterizing efficient attacks and protocols and negligible probabilities allows a clean and well-defined theory. However, it does have some drawbacks:

- It is not really possible in this approach to quantify the security of protocols of fixed size, only of protocols where there is an adjustable security parameter s.
- While any exponential will always beat any polynomial *eventually*, for large enough parameters, any real protocol has specific numerical values assigned and the polynomial might be bigger for those specific values.

Therefore, to evaluate any protocol for real-world use, you must plug in real numbers to see if the security is sufficiently good in practice.